FORM PTO-1390 (Modified)
(REV 11-98)

U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE

10 SEP 2001

TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371

ATTORNEY'S DOCKET NUMBER: 112740-319

U.S. APPLICATION NO. (IF KNOWN, SEE 37 CFR: 09/936385

INTERNATIONAL APPLICATION NO.	INTERNATIONAL FILING DATE	PRIORITY DATE CLAIMED
PCT/DE00/00623	01 March 2000	09 March 1999

TITLE OF INVENTION
METHOD FOR ALLOCATION OF A QUALITY OF SERVICE FOR A PACKET STREAM

APPLICANT(S) FOR DO/EO/US
Michael Wagner

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. ☒ This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.
2. ☐ This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.
3. ☒ This is an express request to begin national examination procedures (35 U.S.C. 371(f)) at any time rather than delay examination until the expiration of the applicable time limit set in 35 U.S.C. 371(b) and PCT Articles 22 and 39(1).
4. ☒ A proper Demand for International Preliminary Examination was made by the 19th month from the earliest claimed priority date.
5. ☒ A copy of the International Application as filed (35 U.S.C. 371 (c) (2))
   - a. ☒ is transmitted herewith (required only if not transmitted by the International Bureau).
   - b. ☐ has been transmitted by the International Bureau.
   - c. ☐ is not required, as the application was filed in the United States Receiving Office (RO/US).
6. ☒ A translation of the International Application into English (35 U.S.C. 371(c)(2)).
7. ☒ A copy of the International Search Report (PCT/ISA/210).
8. ☒ Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371 (c)(3))
   - a. ☒ are transmitted herewith (required only if not transmitted by the International Bureau).
   - b. ☐ have been transmitted by the International Bureau.
   - c. ☐ have not been made; however, the time limit for making such amendments has NOT expired.
   - d. ☐ have not been made and will not be made.
9. ☒ A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).
10. ☐ An oath or declaration of the inventor(s) (35 U.S.C. 371 (c)(4)).
11. ☒ A copy of the International Preliminary Examination Report (PCT/IPEA/409).
12. ☐ A translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371 (c)(5)).

Items 13 to 20 below concern document(s) or information included:

13. ☐ An Information Disclosure Statement under 37 CFR 1.97 and 1.98.
14. ☐ An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included.
15. ☒ A **FIRST** preliminary amendment.
16. ☐ A **SECOND** or **SUBSEQUENT** preliminary amendment.
17. ☒ A substitute specification.
18. ☐ A change of power of attorney and/or address letter.
19. ☒ Certificate of Mailing by Express Mail
20. ☒ Other items or information:

Submission of Drawings - Figures 1-2 on two sheets

PCTUS1/REV03

10 SEP 2001

21. The following fees are submitted:.

BASIC NATIONAL FEE (37 CFR 1.492 (a) (1) - (5)) :

CALCULATIONS | PTO USE ONLY

- ☐ Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2) paid to USPTO and International Search Report not prepared by the EPO or JPO **$1,000.00**
- ☒ International preliminary examination fee (37 CFR 1.482) not paid to USPTO but Internation Search Report prepared by the EPO or JPO **$860.00**
- ☐ International preliminary examination fee (37 CFR 1.482) not paid to USPTO but international search fee (37 CFR 1.445(a)(2)) paid to USPTO **$710.00**
- ☐ International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provisions of PCT Article 33(1)-(4) **$690.00**
- ☐ International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(1)-(4) **$100.00**

				Calculations	PTO use only
ENTER APPROPRIATE BASIC FEE AMOUNT =				**$860.00**	
Surcharge of **$130.00** for furnishing the oath or declaration later than ☐ 20 ☐ 30 months from the earliest claimed priority date (37 CFR 1.492 (e)).				**$0.00**	
CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE		
Total claims	5 - 20 =	0	x $18.00	**$0.00**	
Independent claims	1 - 3 =	0	x $80.00	**$0.00**	
Multiple Dependent Claims **(check if applicable).**			☐	**$0.00**	
TOTAL OF ABOVE CALCULATIONS =				**$860.00**	
Reduction of 1/2 for filing by small entity, if applicable. Verified Small Entity Statement must also be filed (Note 37 CFR 1.9, 1.27, 1.28) **(check if applicable).** ☐				**$0.00**	
SUBTOTAL =				**$860.00**	
Processing fee of **$130.00** for furnishing the English translation later than ☐ 20 ☐ 30 months from the earliest claimed priority date (37 CFR 1.492 (f)). +				**$0.00**	
TOTAL NATIONAL FEE =				**$860.00**	
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31) **(check if applicable).** ☐				**$0.00**	
TOTAL FEES ENCLOSED =				**$860.00**	
				Amount to be: refunded	$
				charged	$

☒ A check in the amount of **$860.00** to cover the above fees is enclosed.

☐ Please charge my Deposit Account No. in the amount of to cover the above fees. A duplicate copy of this sheet is enclosed.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **02-1818** A duplicate copy of this sheet is enclosed.

NOTE: Where an appropriate time limit under 37 CFR 1.494 or 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed and granted to restore the application to pending status.

SEND ALL CORRESPONDENCE TO:

William E. Vaughan (Reg. No. 39,056)
Bell, Boyd & Lloyd LLC
P.O. Box 1135
Chicago, Illinois 60690

SIGNATURE

William E. Vaughan
NAME

39,056
REGISTRATION NUMBER

September 10, 2001
DATE

09/936385
10 SEP 2001

BOX PCT

IN THE UNITED STATES ELECTED/DESIGNATED OFFICE
OF THE UNITED STATES PATENT AND TRADEMARK OFFICE
UNDER THE PATENT COOPERATION TREATY-CHAPTER II

SUBMISSION OF DRAWINGS

APPLICANT:	Michael Wagner	DOCKET NO.:	112740-319
SERIAL NO:		GROUP ART UNIT:	
FILED:		EXAMINER:	
INTERNATIONAL APPLICATION NO.		PCT/DE00/00623	
INTERNATIONAL FILING DATE:		01 March 2000	
INVENTION:	METHOD FOR ALLOCATION OF A QUALITY OF SERVICE FOR A PACKET STREAM		

Assistant Commissioner for Patents,
Washington, D.C. 20231

Sir:

Applicant herewith submits two sheets (Figs. 1-2) of drawings for the above-referenced PCT application.

Respectfully submitted,

(Reg. No. 39,056)

William E. Vaughan
Bell, Boyd & Lloyd LLC
P.O. Box 1135
Chicago, Illinois 60690-1135
(312) 807-4292
Attorneys for Applicant

BOX PCT

IN THE UNITED STATES ELECTED/DESIGNATED OFFICE
OF THE UNITED STATES PATENT AND TRADEMARK OFFICE
UNDER THE PATENT COOPERATION TREATY-CHAPTER II

PRELIMINARY AMENDMENT

APPLICANT:	Michael Wagner.	DOCKET NO: 112740-319
SERIAL NO:		GROUP ART UNIT:
		EXAMINER:
INTERNATIONAL APPLICATION NO:		PCT/DE00/00623
INTERNATIONAL FILING DATE:		01 March 2000
INVENTION:	METHOD FOR ALLOCATION OF A QUALITY OF SERVICE FOR A PACKET STREAM	

Assistant Commissioner for Patents,
Washington, D.C. 20231

Sir:

Please amend the above-identified International Application before entry into the National stage before the U.S. Patent and Trademark Office under 35 U.S.C. §371 as follows:

In the Specification:

Please replace the Specification of the present application, including the Abstract, with the following Substitute Specification:

S P E C I F I C A T I O N

TITLE OF THE INVENTION

METHOD FOR TRANSMITTING MESSAGES BETWEEN A CLIENT INSTANCE ASSIGNED TO A FIRST PROCESS AND AT LEAST ONE SERVER INSTANCE ASSIGNED TO AT LEAST ONE FURTHER PROCESS WITHIN A DISTRIBUTED SYSTEM

BACKGROUND OF THE INVENTION

Distributed systems preferably play a particular role in contemporary telecommunications systems which are generally multiprocessor systems. A distributed system is characterized, in particular, by the fact that processes can be respectively assigned to different processors, and the processors can, if appropriate, be located at spatially separate platforms in the distributed system. In such a case, one of the most important aspects of the communication between the various processes of a distributed system is the platform transparency. This means that a process which wishes to transmit a message to another process does not need to know the platform on which the other process is running at that particular time. Nowadays, such a complex distributed system must also fulfil a larger number of other requirements. It must, inter alia, prove to be extremely reliable, as flexible as possible and as open as possible to adaptations and expansions. The software of such a complex distributed system therefore must be configured in a highly modular fashion with permanently defined open interfaces to the outside so that the individual modules of software are easily adaptable and particularly re-usable.

In order to be able to comply with the abovementioned requirements, in particular in terms of re-usability of software, the software for such a distributed system is generated using object-oriented design methods and/or interprogramming. However, the allocation, necessary in distributed systems, of objects to one another which are usually assigned to different and possibly concurrently running processes, is not solved to a satisfactory degree. To a certain extent, even a purely object-oriented system design must be broken up into conventional procedural programmer techniques, as a result of which the effect of re-using program parts which is achieved with the object orientation is more or less lost.

At present, the following known approaches are being discussed with regard to the introduction of concurrent running and parallel processing into the world of objects:

- Implicit concurrent running: When implicit concurrent running is implemented, there are two possibilities:

- Passive objects: An asynchronous exchange of messages is converted into a sequential synchronous method call or procedure call. Here, the parallel processing of the objects which communicate with each other is very restricted.
- Active objects: The process is started for each object. This procedure leads to a high level of consumption of resources and is therefore only capable of being implemented with a limited number of objects.

- Explicit concurrent running: Here, either a group of objects (object-related), as described in an article by A. Coutts, J.M. Edwards, Model-Driven Distributed Systems, IEEE Concurrency, July 1997, pp. 55-63, or a plurality of events in a sequence (task-related), as explained in an article by M. Awad, J. Ziegler, A Practical Approach to the Design of Concurrency in Object-Oriented Systems, Software - Practice and Experience, September 1997, Vol. 27(9), pp. 1013-1034, are assigned to a process. If the right-hand half of Figure 3 in the aforesaid article by Awad/Ziegler and Figure 5 in the aforesaid article by Coutts/Edwards are considered, it is apparent, at the interfaces between the objects, some of which at the same time represent interfaces between the processes, that the communication between the objects is carried out both via synchronous method calls and via interprocess communication in the form of the asynchronous passing on of messages. Such a definition of the method of communication at the interfaces of objects has the disadvantage that it is made considerably more difficult to re-use and maintain the objects.

Particularly in the context of the communication between various objects of a distributed system, also referred to as instances, which as a rule have what is known as a client/server relationship with one another and which are assigned to various processes, the procedure explained above is a very unfavorable solution in terms of the possibilities of re-use and maintenance which are desired in such a complex system.

A method for converting an interface definition description in an inter-object communications system is already known (EP-A-0 860 776) in which a client object and a server object are provided which are either operated on the same computer or on different computers.

The respective known method is based here on the function of, on the one hand, providing a programming method which is concentrated on origin processing, made available by a server object, while the advantages of a CORBA architecture, that is to say an architecture with a common (object request) broker are to be retained or ensured, and on the other hand of making available an inter-object communications method.

For this purpose there is provision to transmit a message from the client object to the server object in order to execute a specific processing operation. An interface definition conversion program here converts an interface definition description which is written in an interface definition language in order to generate what is referred to as a client stub, a server skeleton and a routing program . The aforementioned interface definition description includes one or more method definition descriptions of data which are necessary for the aforesaid specific processing operation, and a processing result and a message description which specifies a format of the message which is to be output in response to the respective method definition description.

The aforementioned client stub is called in order to cause the client object to output the message. The server skeleton includes a server registration method for storing a routing information item in a routing information memory in order to specify the format of the respective message which can process the server object itself when it is started. In addition, the aforementioned server skeleton includes a method for describing the aforementioned specific processing operation which is to be executed when the server object receives the message.

Finally, the aforementioned routing program decides whether or not the processing of the server object assigned to the routing information item of the

aforementioned message is possible, specifically by comparing the aforementioned message with the aforementioned routing information.

It is also known (EP-A-0 623 876) to allow object managers, which are provided on different computer platforms, to communicate with one another in a cooperative fashion while the objects are enabled to communicate on the respective computer platforms using a remote procedure request.

Furthermore, a method and a device for carrying out efficient CORBA transactions are known (EP-A-0 834 807). However, in the method according to the present invention, procedures are not performed using CORBA transactions.

Finally, further methods for setting up connections between a server and a client are also known (US-A-5 802 367, WO 98 02814 A, GB-A-2 305 270) which, however, solve problems other than those solved by the present invention.

An object of the present invention consists, therefore, in configuring a method for transmitting messages between what is referred to as client and server instances of a distributed system which are respectively assigned to different processes, to the effect that in terms of the implementation of the method, the greatest possible degree of re-use is provided and, at the same time, maintenance is made as easy as possible.

SUMMARY OF THE INVENTION

The object specified above is achieved according to the present invention with a method of the type mentioned at the beginning by virtue of the fact that, after reception of a message directed to at least one server instance by the client instance, a first instance (object handler 1), of partner instances provided as mutual communications partners, which contains the first process selects at least one suitable further instance(object handler 2), containing the at least one further process, of the partner instances, to receive and pass on messages with reference to an allocation table, and by virtue of the fact that the respective further instance (object handler 2) passes on the message to at least one server instance addressed by it~~,~~ and, if appropriate, receives from the at least one server instance a message to be passed on to the client instance via the first instance containing the first process.

The present invention provides the advantage that the definition of the method of communication between the client instance and the at least one server instance is exported into the partner instances which contain a process and which are provided as mutual communications partners. In this way, the messages between the client instance and the first instance containing the first process as well as between the further instance containing at least one server instance and the at least one further process are transmitted synchronously; for example, via a procedure call or method call. The transmission of messages between a first instance containing the first process and a further instance containing at least one further process can then take place asynchronously or synchronously, in a way which is decoupled from the communications interfaces of the client instance and the at least one server instance. As a result, a maximum degree of re-use is achieved especially in terms of the implementation of the client instance and of the at least one server instance. The possibility of maintenance is also considerably improved by virtue of the fact that most communications interfaces between the first instance containing the first process and the further instance containing the at least one further process have to be adapted, but the communications interfaces of the client and of the at least one server instance remain untouched.

A further advantageous embodiment of the present invention provides for the allocation table to contain the type of messages which can be output by the client instance and the address of the further instance containing at least one further process. The type of messages which can be output by the client instance and the address of the further instance which contains at least one further process are therefore entered in this allocation table. This allocation table has the advantage that its contents can be changed at any time and that it is made possible for the first instance containing the first process to make a rapid selection.

According to one embodiment of the present invention, the selection made by the first instance containing the first process can be modified dynamically as a function of the system loading. As a result, the system crashes and blockages in the allocation of the processes to the processors can be avoided.

A further embodiment of the present invention relates to the special case in which the first process and the at least one further process coincide. In this case, the first instance containing the first process and the further instance containing at least one further process are combined in one instance. As a result, the method according to the present invention can be applied to this special case without adaptation.

A further useful embodiment of the present invention can be seen in the method of implementation. For example, all the instances (client instance, server instance, the instance containing the first process and the partner instance) can be implemented in the form of objects whose structure is defined by object classes. Thus, the first instance containing the first process and the further instance containing the at least one further process preferably each having the structure of a common object class. In this way, the principles of purely object-oriented programming are utilized, permitting a high degree of modularity and of re-use and ease of maintenance.

A further-embodiment of the present invention is the very expedient use of the method in a telephone switching system. According to this, all the advantages mentioned above also can be exploited in conjunction with a telephone switching system.

Additional features and advantages of the present invention are described in, and will be apparent from, the following Detailed Description of the Invention and the Figures.

BRIEF DESCRIPTION OF THE FIGURES

Figure 1 shows an exemplary flowchart of the method according to the present invention.

Figure 2 shows an example of an application in the field of a system alarm in a telecommunications system such as a telephone switching system.

A key to the figures is provided at the end of the Detailed Description.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 describes, in a flowchart, the transmission of messages between a client instance assigned to a first process and a server instance assigned to a further

process. The instances of client, server, the first instance containing the first process and the further instance containing the at least one further process as well as the action which is carried out by the server instance are represented in the form of objects with boxes. Accordingly, the object client corresponds to a client instance, the object server to a server instance, the object object handler 1 to a first active instance containing the first process, of the partner instances provided as mutual communications partners, the object object handler 2 to a further active instance containing the further process, of the partner instances, the object action to an action and the object confirm action to an acknowledgement action to a requested action. The active instances which contain the respective processes are characterized here via boxes with bold lines. The type of action is not determined until the specific object action is called.

In the case of an action which is requested by the client and is to be carried out by the server, with an acknowledgement, the method proceeds, for example, as follows:

The client requests from the server an action to which an acknowledgement is to be made. The client calls the action and does not need to know which process is to be carried out or on which processor platform the action is to be carried out. The object handler 1 provides the client for this purpose with the call procedure invoke_action. After the call of the procedure invoke_action, also referred to as method in the object-oriented programming, a uniquely defined number (get handle number) is assigned to the object handler 1 and a timer is started (start timer) which initiates error handling if the acknowledgement is not received at the correct time. Then, the object handler 1 looks for a partner instance provided as communications partner, for example object handler 2 (find target object handler), which is assigned to the action as a function of the type of action, and transmits the message of the action request action_request to the object handler 2. The object handler 2 receives the message, stores the address of its communications partner object handler 1 (store communication partner) together with the number which is uniquely assigned to the object handler 1 and executes the procedure of the object action (execute).

The object action subsequently causes the server addressed by the client to execute the action via the procedure call action. After the execution of the action, the server transmits, in an analogous fashion, an acknowledgement indirectly back to the client. According to this, the following procedure calls, message transmissions and actions run from the server in the direction of the client. The procedure call invoke_action, deletes the address of the communications partner and transmits the action request message for the acknowledgement action_request of the object handler 2 to the object handler 1, which is known to the object handler 2 on the basis of the assigned number, the object handler 1 deletes the assigned number (release handle number) and stops the timer (stop timer), in order to transmit the acknowledgement object handler 1 calls the procedure execute of the object Confirm action, and finally Object Confirm Action executes the procedure confirm_action of the client.

In the case of an action of the server requested by the client being without an acknowledgement, the method according to the present invention of the transmission of messages from the client to the server runs in a similar fashion to that described above. The sequence steps get handle number, start timer, store communication partner and the steps relating to the acknowledgement from the server in the direction of the client are eliminated.

In the case of what is referred to as a broadcast, i.e. a client requests an action from a number of servers, there are various possibilities:

- if the servers addressed by the client are assigned to a common process, the object handler 1 will pass on the action_request message either to an object handler 2, and the object handler 2 ensures that the action is executed by a number of servers, or the object handler 1 transmits a number of action_request messages to a number of object handlers 2 containing the server process, which each cause the servers to execute the action. A combination of both aforesaid variants is also possible.
- if the servers addressed by the client are assigned to different processes, the object handler 1 will in each case transmit an action_request message to the

object handlers 2 containing the different processes, and the object handlers 2 in each case cause the servers to execute the action.

Here too, all combinations of the aforesaid possibilities are conceivable.

The number of actions usually have to be carried out in a distributed system so that each server also can act as a client and each client also can act as a server, and can be combined in an object client and server function.

The advantageous decoupling of process interfaces from the object interfaces of the client and of the server is apparent from the fact that the communication between the client and the server is implemented synchronously via procedure calls and method calls and only the passing on of messages between the object handler 1 and the object handler 2 is, if appropriate, carried out asynchronously without limitation to the process limits.

In the special case in which the client and the server, which are located, for example, on a common platform, can be assigned to the same process, the objects object handler 1 and object handler 2 are combined to form a single object. According to Figure 1, the object handler 1 transmits the action_request message to itself.

Figure 2 shows an application example in the field of a system alarm in a telecommunications system; for example, a telephone switching system.

In a system alarm, there are, for example, the following objects which can act simultaneously as client and server and can request different actions from one another. In addition, the objects can be located on different platforms.

An object Alarm Balance Monitor (ABM) has the function of forming an alarm balance over all the alarms of the instances (AMOI) which are monitored by it and for which alarms can be given. In order to be able to form the alarm balance, the Alarm Balance Monitor requires what is referred to as a SIBS object which is located on a processor platform and provides it with collected information relating to the monitored instances for which alarms can be given.

The boxes constitute the objects Caller, AMOI (AlarmManagedObjectInstance), SIBS (SiteBalanceSupply) and ABM

(AlarmBalanceMonitor). The arrows whose type is not given in the key in the index indicate the message transmission, if appropriate, without limitation to process boundaries, between the objects. The transmission of messages correspond here to the transmission of messages between client and server described in Figure 1. Thus, for example, the Caller object can act as a client and the AMOI object as a server. The same also applies to the other objects AMOI and SIBS as well as SIBS and ABM.

After a system alarm call set_alarm, the following sequence of actions is triggered, for example:

- Set_alarm: a monitored instance AMOI for which an alarm can be given receives a new alarm from a caller, checks the parameters (check_params) determining the alarm and creates a new alarm instance (create contained alarm).
- Confirm: an acknowledgement from the instance (AMOI) to the instance Caller after the system alarm call set_alarm.
- Balance SIBS: at least one server object SIBS is requested to collect the received information necessary for the alarm balance (accumulate alarm status of all associated AMOI).
- Balance ABM: after this the server object ABM is requested to collect the information received from the at least one SIBS object for the alarm balance (accumulate alarm status of all associated SIBS).

Because the actions are requested without limitation to process boundaries, the messages are transmitted from one object to a further object via an active first instance and via a further active partner instance, for example via the object handler 1 and via the object handler 2 from Figure 1, neither of which is illustrated in Figure 2.

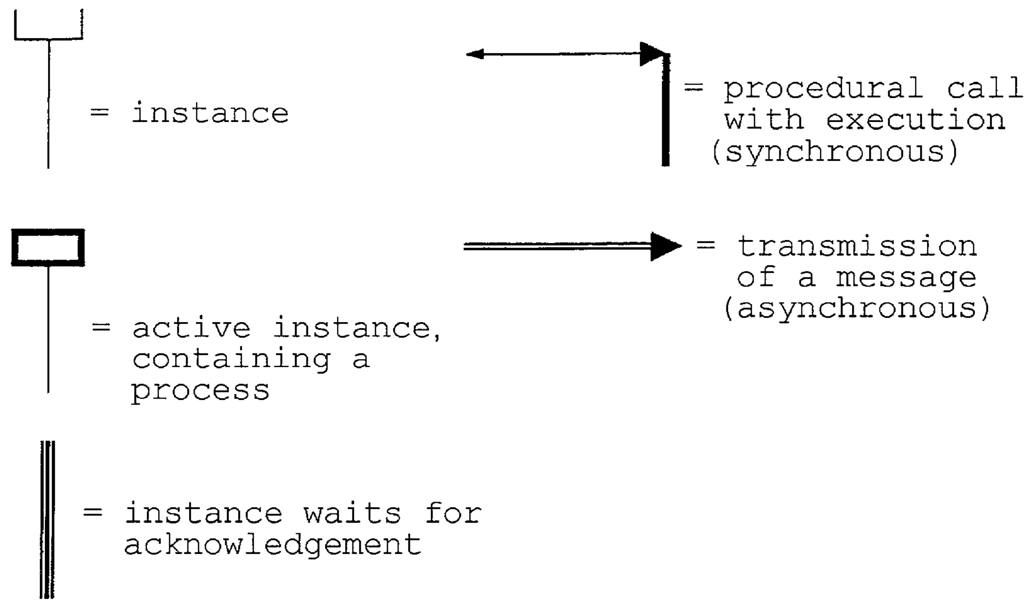
The selection of the object handler 2 made by the object handler 1 can be performed via an allocation table. The allocation table looks, for example, as follows:

Action	Object handler	Confirmation
Set_alarm AMOI	on AMOI platform	yes
Balance SIBS	on SIBS platform	no
Balance ABM	on main platform	no

If a specific action can be executed by different server objects, the allocation of the object handler 2 can be modified as a function of the system load factor.

Although the present invention has been described with reference to specific embodiments, those of skill in the art will recognize that changes may be made thereto without departing from the spirit and scope of the invention as set forth in the hereafter appended claims.

Key to the figures:



ABSTRACT OF THE DISCLOSURE

A method for transmitting messages between a client instance assigned to a first process and at least one server instance assigned to at least one further process within a distributed system, wherein a first instance containing a first process, of partner instances provided as mutual communications partners, selects, after reception of a message directed to at least one server instance by the client instance at least one suitable further instance containing the at least one further process, of the partner instances for the reception and passing on of messages. The at least one further instance containing the at least one further process passes on this message to at least one server instance addressed by it and receives, if appropriate, from the at least server instance, a message to be passed on to the client instance via the first instance containing the first process.

<u>In the Claims:</u>

On page 13, cancel line 1, and substitute the following left-hand justified heading therefor:

CLAIMS

Please cancel claims 1-6, without prejudice, and substitute the following claims therefor:

7. A method for transmitting messages between a client instance assigned to a first process and at least one server instance assigned to a further process within a distributed system, the method comprising the steps of:

receiving a message directed from the client instance to the at least one server instance;

selecting, via a first instance containing the first process, from partner instances provided as mutual communications partners, at least one suitable further instance, containing the further process, of the partner instances for receiving and passing on of messages via an allocation table between a type of messages which can be output by the client instance and an address of the further instance which contains at least one further process;

passing on a message, via the respective further instance, to at least one server instance addressed by it; and

receiving, if appropriate, and at the respective further instance, from the at least one server instance a message to be passed on to the client instance via the first instance containing the first process.

8. A method for transmitting messages as claimed in claim 7, the method further comprising the step of:

modifying dynamically the selection made by the first instance containing the first process as a function of a system load factor.

9. A method for transmitting messages as claimed in claim 7, the method further comprising the step of:

combining in one instance the first instance containing the first process and the further instance containing the at least one further process if the first process and the at least one further process coincide.

10. A method for transmitting messages as claimed in claim 7, wherein all of the instances are objects whose structure is defined by object classes.

11. A method for transmitting messages as claimed in claim 7, wherein the method is applied to a telephone switching system.

REMARKS

The present amendment makes editorial changes and corrects typographical errors in the specification, which includes the Abstract, in order to conform the specification to the requirements of United States Patent Practice. No new matter is added thereby. Attached hereto is a marked-up version of the changes made to the specification by the present amendment. The attached page is captioned "**Version With Markings To Show Changes Made".**

In addition, the present amendment cancels original claims 1-6 in favor of new claims 7-11. Claims 7-11 have been presented solely because the revisions by crossing out and underlining which would have been necessary in claims 1-6 in order to present those claims in accordance with preferred United States Patent Practice would have been too extensive, and thus would have been too burdensome. The present amendment is intended for clarification purposes only and not for substantial reasons related to patentability pursuant to 35 U.S.C. §§103, 102, 103 or 112. Indeed, the cancellation of claims 1-6 does not constitute an intent on the part of the Applicants to surrender any of the subject matter of claims 1-6.

Early consideration on the merits is respectfully requested.

Respectfully submitted,

(Reg. No. 39,056)

William E. Vaughan
Bell, Boyd & Lloyd LLC
P.O. Box 1135
Chicago, Illinois 60690-1135
(312) 807-4292
Attorneys for Applicant

09/936385
2 lf2r5
JC03 Rec'd PCT/PTO 10 SEP 2001

Description

Method for transmitting messages between a client instance assigned to a first process and at least one server instance assigned to at least one further process within a distributed system

The invention relates to a method for transmitting messages between a client instance (client) assigned to a first process and at least one server instance (server) assigned to at least one further process within a distributed system.

Distributed systems preferably play a particular role in contemporary telecommunications systems which are generally multiprocessor systems. A distributed system is characterized in particular by the fact that processes can be respectively assigned to different processors, and the processors can, if appropriate, be located at spatially separate platforms in the distributed system. In such a case, one of the most important aspects of the communication between the various processes of a distributed system is the platform transparency. This means that a process which wishes to transmit a message to another process does not need to know the platform on which the other process is running at that particular time. Nowadays, such a complex distributed system must also fulfil a larger number of other requirements. It must, inter alia, prove to be extremely reliable, as flexible as possible and as open as possible to adaptations and expansions. The software of such a complex distributed system must therefore be configured in a highly modular fashion with permanently defined open interfaces to the outside so that the individual modules of software are easily adaptable and particularly re-usable.

In order to be able to comply with the abovementioned requirements, in particular in terms of re-usability of software,

the software for such a distributed system is generated using object-oriented design methods and/or interprogramming. However, the allocation, necessary in distributed systems, of objects to one another which are usually assigned to different and possibly concurrently running processes, is not solved to a satisfactory degree. To a certain extent, even a purely object-oriented system design must be broken up into conventional procedural programmer techniques, as a result of which the effect of re-using program parts which is achieved with the object orientation is more or less lost.

At present, the following known approaches are being discussed with regard to the introduction of concurrent running and parallel processing into the world of objects:

- Implicit concurrent running: When implicit concurrent running is implemented, there are two possibilities:
  - Passive objects: An asynchronous exchange of messages is converted into a sequential synchronous method call or procedure call. Here, the parallel processing of the objects which communicate with each other is very restricted.
  - Active objects: The process is started for each object. This procedure leads to a high level of consumption of resources and is therefore only capable of being implemented with a limited number of objects.

- Explicit concurrent running: Here either a group of objects (object-related), as described in an article by A. Coutts, J.M. Edwards, Model-Driven Distributed Systems, IEEE Concurrency, July 1997, pp. 55-63, or a plurality of events in a sequence (task-related),

as explained in an article by M. Awad, J. Ziegler, A Practical Approach to the Design of Concurrency in Object-Oriented Systems, Software - Practice and Experience,

September 1997, Vol. 27(9), pp. 1013-1034, are assigned to a process. If the right-hand half of figure 3 in the aforesaid article by Awad/Ziegler and figure 5 in the aforesaid article by Coutts/Edwards are considered, it is apparent, at the interfaces between the objects, some of which at the same time represent interfaces between the processes, that the communication between the objects is carried out both by means of synchronous method calls and by means of interprocess communication in the form of the asynchronous passing on of messages. Such a definition of the method of communication at the interfaces of objects has the disadvantage that it is made considerably more difficult to re-use and maintain the objects.

In particular in the context of the communication between various objects of a distributed system, also referred to as instances, which as a rule have what is known as a client/server relationship with one another and which are assigned to various processes, the procedure explained above is a very unfavorable solution in terms of the possibilities of re-use and maintenance which are desired in such a complex system.

A method for converting an interface definition description in an inter-object communications system is already known (EP-A-0 860 776) in which a client object and a server object are provided which are either operated on the same computer or on different computers.

The respective known method is based here on the function of, on the one hand, providing a programming method which is concentrated on origin processing, made available by a server object, while the advantages of a CORBA architecture, that is to say an architecture

with a common (object request) broker are to be retained

or ensured, and on the other hand of making available an inter-object communications method.

For this purpose there is provision to transmit a message from the client object to the server object in order to execute a specific processing operation. An interface definition conversion program here converts an interface definition description which is written in an interface definition language in order to generate what is referred to as a client stub, a server skeleton and a routing prgram. The aforementioned interface definition description comprises one or more method definition descriptions of data which are necessary for the aforesaid specific processing operation, and a processing result and a message description which specifies a format of the message which is to be output in response to the respective method definition description.

The aforementioned client stub is called in order to cause the client object to output the message. The server skeleton comprises a server registration method for storing a routing information item in a routing information memory in order to specify the format of the respective message which can process the server object itself when it is started. In addition, the aforementioned server skeleton comprises a method for describing the aforementioned specific processing operation which is to be executed when the server object receives the message.

Finally, the aforementioned routing program decides whether or not the processing of the server object assigned to the routing information item of the aforementioned message is possible, specifically by comparing the aforementioned message with the aforementioned routing information.

The method according to the present invention cannot be compared with this known procedure.

It is also known (EP-A-0 623 876) to allow object managers, which are provided on different computer platforms, to communicate with one another in a cooperative fashion while the objects are enabled to communicate on the respective computer platforms using a remote procedure request. This procedure also has nothing to do with the method according to the invention.

Furthermore, a method and a device for carrying out efficient CORBA transactions are known (EP-A-0 834 807). However, in the method according to the invention procedures are not performed using CORBA transactions.

Finally, further methods for setting up connections between a server and a client are also known (US-A-5 802 367, WO 98 02814 A, GB-A-2 305 270), which, however, solve problems other than those solved by the present invention.

The object of the invention consists therefore in configuring a method for transmitting messages between what is referred to as client and server instances of a distributed system which are respectively assigned to different processes, to the effect that in terms of the implementation of the method, the greatest possible degree of re-use is provided and at the same time maintenance is made as easy as possible.

The object specified above is achieved according to the invention with a method of the type mentioned at the beginning by virtue of the fact that, after reception of a message directed to at least one server instance by the client instance, a first instance (object handler 1), of partner instances provided as mutual communications partners, which contains the first process selects at least one suitable further instance

(object handler 2), containing the at least one further process, of the partner instances, to receive and pass on messages with reference to an allocation table, and by virtue of the fact that the respective further instance (object handler 2) passes on the message to at least one server instance addressed by it, and, if appropriate, receives from the at least one server instance a message to be passed on to the client instance via the first instance containing the first process.

The invention provides the advantage that the definition of the method of communication between the client instance and the at least one server instance is exported into the partner instances which contain a process and which are provided as mutual communications partners. In this way, the messages between the client instance and the first instance containing the first process as well as between the further instance containing at least one server instance and the at least one further process are transmitted synchronously, for example by means of a procedure call or method call. The transmission of messages between a first instance containing the first process and a further instance containing at least one further process can then take place asynchronously or synchronously, in a way which is decoupled from the communications interfaces of the client instance and the at least one server instance. As a result, a maximum degree of re-use is achieved especially in terms of the implementation of the client instance and of the at least one server instance. The possibility of maintenance is also considerably improved by virtue of the fact that most communications interfaces between the first instance containing the first process and the further instance containing the at least one further process have to be adapted, but the communications

interfaces of the client and of the at least one server instance remain untouched.

A further advantageous refinement of the invention provides for the allocation table to contain the type of messages which can be output by the client instance and the address of the further instance containing at least one further process. The type of messages which can be output by the client instance and the address of the further instance which contains at least one further process are therefore entered in this allocation table. This allocation table has the advantage that its contents can be changed at any time and that it is made possible for the first instance containing the first process to make a rapid selection.

According to one expedient development of the invention, the selection made by the first instance containing the first process can be modified dynamically as a function of the system loading. As a result, the system crashes and blockages in the allocation of the processes to the processors can be avoided.

A further refinement of the invention relates to the special case in which the first process and the at least one further process coincide. In this case, the first instance containing the first process and the further instance containing at least one further process are combined in one instance. As a result, the method according to the invention can be applied to this special case without adaptation.

A further useful refinement of the invention can be seen in the method of implementation. For example, all the instances (client instance, server instance, the instance containing the first process and the partner instance) can be implemented in the form of objects whose

object class. In this way, the principles of purely object-oriented programming are utilized, permitting a high degree of modularity and of re-use and ease of maintenance.

A further refinement of the invention is the very expedient use of the method according to the invention in a telephone switching system. According to this, all the advantages mentioned above can also be exploited in conjunction with a telephone switching system.

An exemplary embodiment of the invention is described in more detail below with reference to a drawing, in which:

Figure 1 shows an exemplary flowchart of the method according to the invention,

Figure 2 shows an example of an application in the field of a system alarm in a telecommunications system such as a telephone switching system.

A key to the figures is provided in an annex at the end of the description.

Figure 1 describes, in a flowchart, the transmission of messages between a client instance assigned to a first process and a server instance assigned to a further process. The instances of client, server, the first instance containing the first process and the further instance containing the at least one further process as well as the action which is carried out by the server instance are represented in the form of objects with boxes. Accordingly, the object client corresponds to a client instance, the object server to a server instance, the object object handler 1 to a first active instance containing the first process, of the

partner instances provided as mutual communications partners, the object object handler 2 to a further active instance containing the further process, of the partner instances, the object action to an action and the object confirm action to an acknowledgement action to a requested action. The active instances which contain the respective processes are characterized here by means of boxes with bold lines. The type of action is not determined until the specific object action is called.

In the case of an action which is requested by the client and is to be carried out by the server, with an acknowledgement, the method proceeds, for example, as follows:

The client requests from the server an action to which an acknowledgement is to be made. The client calls the action and does not need to know which process is to be carried out or on which processor platform the action is to be carried out. The object handler 1 provides the client for this purpose with the call procedure invoke_action. After the call of the procedure invoke_action, also referred to as method in the object-oriented programming, a uniquely defined number (get handle number) is assigned to the object handler 1 and a timer is started (start timer) which initiates error handling if the acknowledgement is not received at the correct time. Then, the object handler 1 looks for a partner instance provided as communications partner, for example object handler 2 (find target object handler), which is assigned to the action as a function of the type of action, and transmits the message of the action request action_request to the object handler 2. The object handler 2 receives the message, stores the address of its communications partner object handler 1 (store communication partner) together with the number which is uniquely assigned to the object handler 1 and executes the procedure

of the object action (execute). The object action subsequently causes the server addressed by the client to execute the action by means of the procedure call action. After the execution of the action, the

server transmits, in an analogous fashion, an acknowledgement indirectly back to the client. According to this, the following procedure calls, message transmissions and actions run from the server in the direction of the client. The procedure call invoke_action, deletes the address of the communications partner and transmits the action request message for the acknowledgement action_request of the object handler 2 to the object handler 1, which is known to the object handler 2 on the basis of the assigned number, the object handler 1 deletes the assigned number (release handle number) and stops the timer (stop timer), in order to transmit the acknowledgement object handler 1 calls the procedure execute of the object Confirm action, and finally Object Confirm Action executes the procedure confirm_action of the client.

In the case of an action of the server requested by the client being without an acknowledgement, the method according to the invention of the transmission of messages from the client to the server runs in a similar fashion to that described above. The sequence steps get handle number, start timer, store communication partner and the steps relating to the acknowledgement from the server in the direction of the client are eliminated.

In the case of what is referred to as a broadcast, i.e. a client requests an action from a plurality of servers, there are various possibilities:

- if the servers addressed by the client are assigned to a common process, the object handler 1 will pass on the action_request message either to an object handler 2, and the object handler 2 ensures that the action is executed by a plurality of servers, or the object handler 1 transmits a plurality of action_request messages to a plurality of object handlers 2 containing the

server process, which each cause the servers to execute the action. A combination of both aforesaid variants is also possible.

- if the servers addressed by the client are assigned to different processes, the object handler 1 will in each case

transmit an action_request message to the object handlers 2 containing the different processes, and the object handlers 2 in each case cause the servers to execute the action.

Here too, all combinations of the aforesaid possibilities are conceivable.

The plurality of actions usually have to be carried out in a distributed system so that of course each server can also act as a client and each client can also act as a server, and can be combined in an object client and server function.

The advantageous decoupling of process interfaces from the object interfaces of the client and of the server is apparent from the fact that the communication between the client and the server is implemented synchronously by means of procedure calls and method calls and only the passing on of messages between the object handler 1 and the object handler 2 is, if appropriate, carried out asynchronously without limitation to the process limits.

In the special case in which the client and the server which are located, for example, on a common platform, can be assigned to the same process, the objects object handler 1 and object handler 2 are combined to form a single object. According to Figure 1, in this case the object handler 1 transmits the action_request message to itself.

Figure 2 shows an application example in the field of a system alarm in a telecommunications system, for example a telephone switching system.

In a system alarm, there are, for example, the following objects which can act simultaneously as client and server and can request different actions

from one another. In addition,

the objects can be located on different platforms.

An object Alarm Balance Monitor (ABM) has the function of forming an alarm balance over all the alarms of the instances (AMOI) which are monitored by it and for which alarms can be given. In order to be able to form the alarm balance, the Alarm Balance Monitor requires what is referred to as a SIBS object which is located on a processor platform and provides it with collected information relating to the monitored instances for which alarms can be given.

The boxes constitute the objects Caller, AMOI (AlarmManagedObjectInstance), SIBS (SiteBalanceSupply) and ABM (AlarmBalanceMonitor). The arrows whose type is not given in the key in the index indicate the message transmission, if appropriate, without limitation to process boundaries, between the objects. The transmission of messages correspond here to the transmission of messages between client and server described in Figure 1. Thus, for example the Caller object can act as a client and the AMOI object as a server. The same also applies to the other objects AMOI and SIBS as well as SIBS and ABM.

After a system alarm call set_alarm, the following sequence of actions is triggered, for example:

- Set_alarm: a monitored instance AMOI for which an alarm can be given receives a new alarm from a caller, checks the parameters (check_params) determining the alarm and creates a new alarm instance (create contained alarm).
- Confirm: an acknowledgement from the instance (AMOI) to the instance Caller after the system alarm call set_alarm.
- Balance SIBS: at least one server object SIBS is requested to collect the received information necessary for the alarm balance (accumulate alarm status of all associated AMOI).

- Balance ABM: after this the server object ABM is requested

to collect the information received from the at least one SIBS object for the alarm balance (accumulate alarm status of all associated SIBS).

Because the actions are requested without limitation to process boundaries, the messages are transmitted from one object to a further object via an active first instance and via a further active partner instance, for example via the object handler 1 and via the object handler 2 from Figure 1, neither of which is illustrated in Figure 2.

The selection of the object handler 2 made by the object handler 1 can be performed by means of an allocation table. The allocation table looks, for example, as follows:

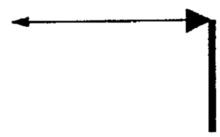
Action	Object handler	Confirmation
Set_alarm AMOI	on AMOI platform	yes
Balance SIBS	on SIBS platform	no
Balance ABM	on main platform	no

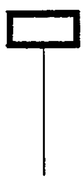
If a specific action can be executed by different server objects, the allocation of the object handler 2 can be modified as a function of the system load factor.

APPENDIX

Key to the figures:


= instance


= procedural call with execution (synchronous)


= active instance, containing a process


= transmission of a message (asynchronous)


= instance waits for acknowledgement

New Patent claims
(replace the previously applicable patent claims)

1. A method for transmitting messages between a client instance (client) assigned to a first process and at least one server instance (server) assigned to a further process within a distributed system, characterized in that, after reception of a message directed from the client instance to at least one server instance, a first instance (object handler 1) containing the first process selects, from partner instances provided as mutual communications partners, at least one suitable further instance (object handler 2), containing the at least one further process, of the partner instances for the reception and passing on of messages by means of an allocation table between the type of messages which can be output by the client instance and the address of the further instance and the address of the further instance containing at least one further process, and in that the respective further instance (object handler 2) passes on the message to at least one server instance addressed by it, and if appropriate, receives from the at least one server instance a message to be passed on to the client instance via the first instance containing the first process.

2. The method as claimed in claim 1, characterized in that the selection made by the first instance containing the first process can be modified dynamically as a function of the system load factor.

3. The method as claimed in one of the preceding claims, characterized in that if the first process and the at least one further process coincide, the first instance containing the first process and

the further instance containing the at least one further process are combined in one instance.

4. The method as claimed in one of the preceding claims, characterized in that all the instances are implemented in the form of objects whose structure is defined by object classes.

5. The method as claimed in one of the preceding claims, characterized in that said method is applied to a telephone switching system.

~~Description~~

SPECIFICATION

TITLE OF THE INVENTION

METHOD FOR TRANSMITTING MESSAGES BETWEEN A CLIENT INSTANCE ASSIGNED TO A FIRST PROCESS AND AT LEAST ONE SERVER INSTANCE ASSIGNED TO AT LEAST ONE FURTHER PROCESS WITHIN A DISTRIBUTED SYSTEM

~~Method for transmitting messages between a client instance assigned to a first process and at least one server instance assigned to at least one further process within a distributed system~~

~~The invention relates to a method for transmitting messages between a client instance (client) assigned to a first process and at least one server instance (server) assigned to at least one further process within a distributed system.~~

BACKGROUND OF THE INVENTION

Distributed systems preferably play a particular role in contemporary telecommunications systems which are generally multiprocessor systems. A distributed system is characterized, in particular, by the fact that processes can be respectively assigned to different processors, and the processors can, if appropriate, be located at spatially separate platforms in the distributed system. In such a case, one of the most important aspects of the communication between the various processes of a distributed system is the platform transparency. This means that a process which wishes to transmit a message to another process does not need to know the platform on which the other process is running at that particular time. Nowadays, such a complex distributed system must also fulfil a larger number of other requirements. It must, inter alia, prove to be extremely reliable, as flexible as possible and as open as possible to adaptations and expansions. The software of such a complex distributed system ~~must~~ therefore must be configured in a highly modular fashion with permanently defined open interfaces to the outside so that the individual modules of software are easily adaptable and particularly re-usable.

In order to be able to comply with the abovementioned requirements, in particular in terms of re-usability of software, the software for such a distributed system is generated using object-oriented design methods and/or interprogramming. However, the allocation, necessary in distributed systems, of objects to one another which are usually assigned to different and possibly concurrently running processes, is not solved to a satisfactory degree. To a certain extent, even a purely object-oriented system design must be broken up into conventional procedural programmer techniques, as a result of which the effect of re-using program parts which is achieved with the object orientation is more or less lost.

At present, the following known approaches are being discussed with regard to the introduction of concurrent running and parallel processing into the world of objects:

- Implicit concurrent running: When implicit concurrent running is implemented, there are two possibilities:
  - Passive objects: An asynchronous exchange of messages is converted into a sequential synchronous method call or procedure call. Here, the parallel processing of the objects which communicate with each other is very restricted.
  - Active objects: The process is started for each object. This procedure leads to a high level of consumption of resources and is therefore only capable of being implemented with a limited number of objects.
- Explicit concurrent running: Here, either a group of objects (object-related), as described in an article by A. Coutts, J.M. Edwards, Model-Driven Distributed Systems, IEEE Concurrency, July 1997, pp. 55-63, or a plurality of events in a sequence (task-related), as explained in an article by M. Awad, J. Ziegler, A Practical Approach to the Design of Concurrency in Object-Oriented Systems, Software - Practice and Experience, September 1997, Vol. 27(9), pp. 1013-1034, are assigned to a process. If the right-hand half of ~~figure~~ Figure 3 in the aforesaid article by Awad/Ziegler and ~~figure~~ Figure 5 in the aforesaid article by Coutts/Edwards are considered, it is

apparent, at the interfaces between the objects, some of which at the same time represent interfaces between the processes, that the communication between the objects is carried out both ~~by means of~~ via synchronous method calls and ~~by means of~~ via interprocess communication in the form of the asynchronous passing on of messages. Such a definition of the method of communication at the interfaces of objects has the disadvantage that it is made considerably more difficult to re-use and maintain the objects.

~~In particular~~ Particularly in the context of the communication between various objects of a distributed system, also referred to as instances, which as a rule have what is known as a client/server relationship with one another and which are assigned to various processes, the procedure explained above is a very unfavorable solution in terms of the possibilities of re-use and maintenance which are desired in such a complex system.

A method for converting an interface definition description in an inter-object communications system is already known (EP-A-0 860 776) in which a client object and a server object are provided which are either operated on the same computer or on different computers.

The respective known method is based here on the function of, on the one hand, providing a programming method which is concentrated on origin processing, made available by a server object, while the advantages of a CORBA architecture, that is to say an architecture with a common (object request) broker are to be retained or ensured, and on the other hand of making available an inter-object communications method.

For this purpose there is provision to transmit a message from the client object to the server object in order to execute a specific processing operation. An interface definition conversion program here converts an interface definition description which is written in an interface definition language in order to generate what is referred to as a client stub, a server skeleton and a routing ~~prgram program~~ . The aforementioned interface definition description ~~comprises~~ includes one or more method definition descriptions of data which are necessary for the aforesaid specific

processing operation, and a processing result and a message description which specifies a format of the message which is to be output in response to the respective method definition description.

The aforementioned client stub is called in order to cause the client object to output the message. The server skeleton ~~comprises~~ includes a server registration method for storing a routing information item in a routing information memory in order to specify the format of the respective message which can process the server object itself when it is started. In addition, the aforementioned server skeleton ~~comprises~~ includes a method for describing the aforementioned specific processing operation which is to be executed when the server object receives the message.

Finally, the aforementioned routing program decides whether or not the processing of the server object assigned to the routing information item of the aforementioned message is possible, specifically by comparing the aforementioned message with the aforementioned routing information.

~~The method according to the present invention cannot be compared with this known procedure.~~

It is also known (EP-A-0 623 876) to allow object managers, which are provided on different computer platforms, to communicate with one another in a cooperative fashion while the objects are enabled to communicate on the respective computer platforms using a remote procedure request. ~~This procedure also has nothing to do with the method according to the invention.~~

Furthermore, a method and a device for carrying out efficient CORBA transactions are known (EP-A-0 834 807). However, in the method according to the present invention, procedures are not performed using CORBA transactions.

Finally, further methods for setting up connections between a server and a client are also known (US-A-5 802 367, WO 98 02814 A, GB-A-2 305 270)~~,~~ which, however, solve problems other than those solved by the present invention.

~~The~~ An object of the present invention consists, therefore, in configuring a method for transmitting messages between what is referred to as client and server instances of a distributed system which are respectively assigned to different

processes, to the effect that in terms of the implementation of the method, the greatest possible degree of re-use is provided and, at the same time, maintenance is made as easy as possible.

SUMMARY OF THE INVENTION

The object specified above is achieved according to the present invention with a method of the type mentioned at the beginning by virtue of the fact that, after reception of a message directed to at least one server instance by the client instance, a first instance (object handler 1), of partner instances provided as mutual communications partners, which contains the first process selects at least one suitable further instance(object handler 2), containing the at least one further process, of the partner instances, to receive and pass on messages with reference to an allocation table, and by virtue of the fact that the respective further instance (object handler 2) passes on the message to at least one server instance addressed by it~~,~~ and, if appropriate, receives from the at least one server instance a message to be passed on to the client instance via the first instance containing the first process.

The present invention provides the advantage that the definition of the method of communication between the client instance and the at least one server instance is exported into the partner instances which contain a process and which are provided as mutual communications partners. In this way, the messages between the client instance and the first instance containing the first process as well as between the further instance containing at least one server instance and the at least one further process are transmitted synchronously~~,~~; for example, ~~by means of~~ via a procedure call or method call. The transmission of messages between a first instance containing the first process and a further instance containing at least one further process can then take place asynchronously or synchronously, in a way which is decoupled from the communications interfaces of the client instance and the at least one server instance. As a result, a maximum degree of re-use is achieved especially in terms of the implementation of the client instance and of the at least one server instance. The possibility of maintenance is also considerably improved by virtue of the fact that most communications interfaces between the first instance

containing the first process and the further instance containing the at least one further process have to be adapted, but the communications interfaces of the client and of the at least one server instance remain untouched.

A further advantageous ~~refinement~~ embodiment of the present invention provides for the allocation table to contain the type of messages which can be output by the client instance and the address of the further instance containing at least one further process. The type of messages which can be output by the client instance and the address of the further instance which contains at least one further process are therefore entered in this allocation table. This allocation table has the advantage that its contents can be changed at any time and that it is made possible for the first instance containing the first process to make a rapid selection.

According to one ~~expedient development~~ embodiment of the present invention, the selection made by the first instance containing the first process can be modified dynamically as a function of the system loading. As a result, the system crashes and blockages in the allocation of the processes to the processors can be avoided.

A further ~~refinement~~ embodiment of the present invention relates to the special case in which the first process and the at least one further process coincide. In this case, the first instance containing the first process and the further instance containing at least one further process are combined in one instance. As a result, the method according to the present invention can be applied to this special case without adaptation.

A further useful ~~refinement~~ embodiment of the present invention can be seen in the method of implementation. For example, all the instances (client instance, server instance, the instance containing the first process and the partner instance) can be implemented in the form of objects whose structure is defined by object classes. Thus, the first instance containing the first process and the further instance containing the at least one further process preferably each having the structure of a common object class. In this way, the principles of purely

object-oriented programming are utilized, permitting a high degree of modularity and of re-use and ease of maintenance.

A further ~~refinement~~ embodiment of the present invention is the very expedient use of the method ~~according to the invention~~ in a telephone switching system. According to this, all the advantages mentioned above ~~can~~ also can be exploited in conjunction with a telephone switching system.

Additional features and advantages of the present invention are described in, and will be apparent from, the following Detailed Description of the Invention and the Figures.

~~An exemplary embodiment of the invention is described in more detail below with reference to a drawing, in which:~~

BRIEF DESCRIPTION OF THE FIGURES

Figure 1 shows an exemplary flowchart of the method according to the present invention~~,~~.

Figure 2 shows an example of an application in the field of a system alarm in a telecommunications system such as a telephone switching system.

A key to the figures is provided at the end of the Detailed Description.

~~A key to the figures is provided in an annex at the end of the description.~~

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 describes, in a flowchart, the transmission of messages between a client instance assigned to a first process and a server instance assigned to a further process. The instances of client, server, the first instance containing the first process and the further instance containing the at least one further process as well as the action which is carried out by the server instance are represented in the form of objects with boxes. Accordingly, the object client corresponds to a client instance, the object server to a server instance, the object object handler 1 to a first active instance containing the first process, of the partner instances provided as mutual communications partners, the object object handler 2 to a further active instance containing the further process, of the partner instances, the object action to an action and the object confirm action to an acknowledgement action to a requested

action. The active instances which contain the respective processes are characterized here ~~by means of~~ via boxes with bold lines. The type of action is not determined until the specific object action is called.

In the case of an action which is requested by the client and is to be carried out by the server, with an acknowledgement, the method proceeds, for example, as follows:

The client requests from the server an action to which an acknowledgement is to be made. The client calls the action and does not need to know which process is to be carried out or on which processor platform the action is to be carried out. The object handler 1 provides the client for this purpose with the call procedure invoke_action. After the call of the procedure invoke_action, also referred to as method in the object-oriented programming, a uniquely defined number (get handle number) is assigned to the object handler 1 and a timer is started (start timer) which initiates error handling if the acknowledgement is not received at the correct time. Then, the object handler 1 looks for a partner instance provided as communications partner, for example object handler 2 (find target object handler), which is assigned to the action as a function of the type of action, and transmits the message of the action request action_request to the object handler 2. The object handler 2 receives the message, stores the address of its communications partner object handler 1 (store communication partner) together with the number which is uniquely assigned to the object handler 1 and executes the procedure of the object action (execute). The object action subsequently causes the server addressed by the client to execute the action ~~by means of~~ via the procedure call action. After the execution of the action, the server transmits, in an analogous fashion, an acknowledgement indirectly back to the client. According to this, the following procedure calls, message transmissions and actions run from the server in the direction of the client. The procedure call invoke_action, deletes the address of the communications partner and transmits the action request message for the acknowledgement action_request of the object handler 2 to the object handler 1, which is known to the object handler 2 on the basis of the assigned number, the object handler 1 deletes

the assigned number (release handle number) and stops the timer (stop timer), in order to transmit the acknowledgement object handler 1 calls the procedure execute of the object Confirm action, and finally Object Confirm Action executes the procedure confirm_action of the client.

In the case of an action of the server requested by the client being without an acknowledgement, the method according to the present invention of the transmission of messages from the client to the server runs in a similar fashion to that described above. The sequence steps get handle number, start timer, store communication partner and the steps relating to the acknowledgement from the server in the direction of the client are eliminated.

In the case of what is referred to as a broadcast, i.e. a client requests an action from a ~~plurality~~ number of servers, there are various possibilities:

- if the servers addressed by the client are assigned to a common process, the object handler 1 will pass on the action_request message either to an object handler 2, and the object handler 2 ensures that the action is executed by a ~~plurality~~ number of servers, or the object handler 1 transmits a ~~plurality~~ number of action_request messages to a ~~plurality~~ number of object handlers 2 containing the server process, which each cause the servers to execute the action. A combination of both aforesaid variants is also possible.
- if the servers addressed by the client are assigned to different processes, the object handler 1 will in each case transmit an action_request message to the object handlers 2 containing the different processes, and the object handlers 2 in each case cause the servers to execute the action.

Here too, all combinations of the aforesaid possibilities are conceivable.

The ~~plurality~~ number of actions usually have to be carried out in a distributed system so that ~~of course~~ each server ~~can~~ also can act as a client and each client ~~can~~ also can act as a server, and can be combined in an object client and server function.

The advantageous decoupling of process interfaces from the object interfaces of the client and of the server is apparent from the fact that the

communication between the client and the server is implemented synchronously ~~by means of~~ via procedure calls and method calls and only the passing on of messages between the object handler 1 and the object handler 2 is, if appropriate, carried out asynchronously without limitation to the process limits.

In the special case in which the client and the server, which are located, for example, on a common platform, can be assigned to the same process, the objects object handler 1 and object handler 2 are combined to form a single object. According to Figure 1, ~~in this case~~ the object handler 1 transmits the action_request message to itself.

Figure 2 shows an application example in the field of a system alarm in a telecommunications system~~,~~; for example, a telephone switching system.

In a system alarm, there are, for example, the following objects which can act simultaneously as client and server and can request different actions from one another. In addition, the objects can be located on different platforms.

An object Alarm Balance Monitor (ABM) has the function of forming an alarm balance over all the alarms of the instances (AMOI) which are monitored by it and for which alarms can be given. In order to be able to form the alarm balance, the Alarm Balance Monitor requires what is referred to as a SIBS object which is located on a processor platform and provides it with collected information relating to the monitored instances for which alarms can be given.

The boxes constitute the objects Caller, AMOI (AlarmManagedObjectInstance), SIBS (SiteBalanceSupply) and ABM (AlarmBalanceMonitor). The arrows whose type is not given in the key in the index indicate the message transmission, if appropriate, without limitation to process boundaries, between the objects. The transmission of messages correspond here to the transmission of messages between client and server described in Figure 1. Thus, for example, the Caller object can act as a client and the AMOI object as a server. The same also applies to the other objects AMOI and SIBS as well as SIBS and ABM.

After a system alarm call set_alarm, the following sequence of actions is triggered, for example:

- Set_alarm: a monitored instance AMOI for which an alarm can be given receives a new alarm from a caller, checks the parameters (check_params) determining the alarm and creates a new alarm instance (create contained alarm).
- Confirm: an acknowledgement from the instance (AMOI) to the instance Caller after the system alarm call set_alarm.
- Balance SIBS: at least one server object SIBS is requested to collect the received information necessary for the alarm balance (accumulate alarm status of all associated AMOI).
- Balance ABM: after this the server object ABM is requested to collect the information received from the at least one SIBS object for the alarm balance (accumulate alarm status of all associated SIBS).

Because the actions are requested without limitation to process boundaries, the messages are transmitted from one object to a further object via an active first instance and via a further active partner instance, for example via the object handler 1 and via the object handler 2 from Figure 1, neither of which is illustrated in Figure 2.

The selection of the object handler 2 made by the object handler 1 can be performed ~~by means of~~ via an allocation table. The allocation table looks, for example, as follows:

Action	Object handler	Confirmation
Set_alarm AMOI	on AMOI platform	yes
Balance SIBS	on SIBS platform	no
Balance ABM	on main platform	no

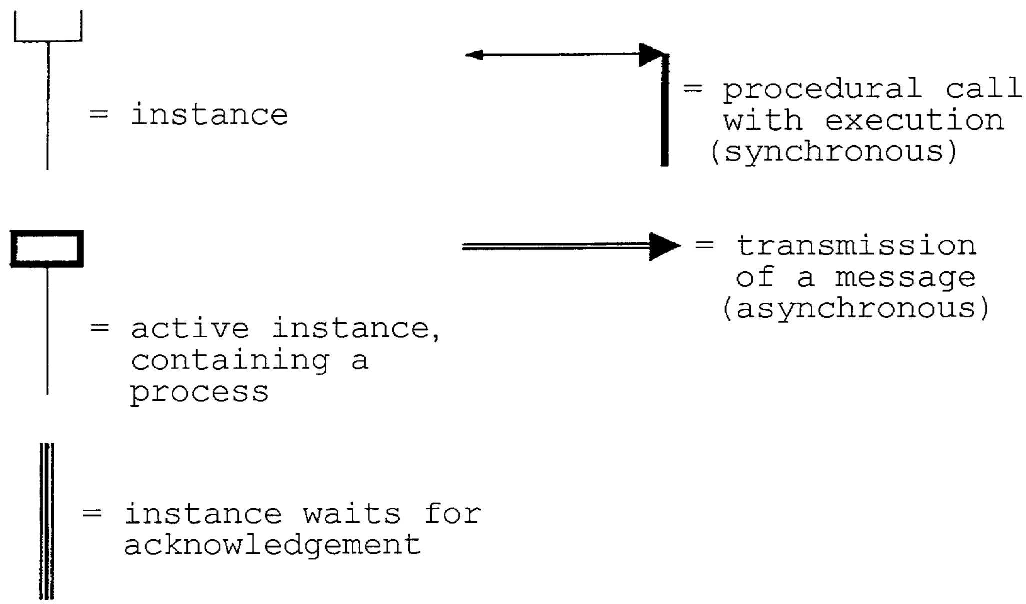
If a specific action can be executed by different server objects, the allocation of the object handler 2 can be modified as a function of the system load factor.

Although the present invention has been described with reference to specific embodiments, those of skill in the art will recognize that changes may be made

thereto without departing from the spirit and scope of the invention as set forth in the hereafter appended claims.

~~APPENDIX~~

Key to the figures:



~~Abstract~~

ABSTRACT OF THE DISCLOSURE

~~Method for transmitting messages between a client instance assigned to a first process and at least one server instance assigned to at least one further process within a distributed system~~

~~A~~ A method for transmitting messages between a client instance assigned to a first process and at least one server instance assigned to at least one further process within a distributed system, wherein a first instance ~~(object handler 1)~~ containing a first process, of partner instances provided as mutual communications partners, selects, after reception of a message directed to at least one server instance ~~(server)~~ by the client instance ~~(client)~~ at least one suitable further instance ~~(object handler 2)~~ containing the at least one further process, of the partner instances for the reception and passing on of messages. The at least one further instance containing the at least one further process passes on this message to at least one server instance addressed by it and receives, if appropriate, from the at least server instance, a message to be passed on to the client instance via the first instance containing the first process.

~~Figure 1~~

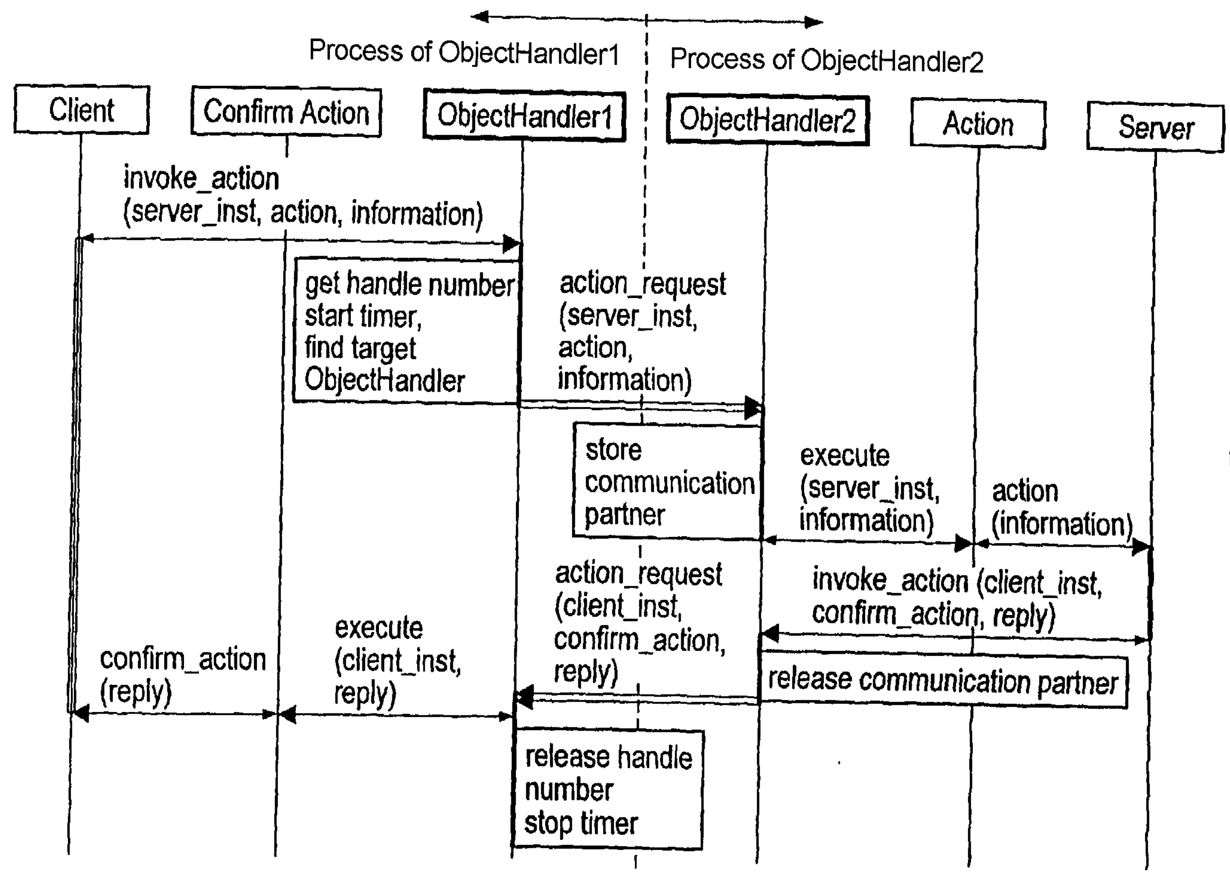

Process of ObjectHandler1
Process of ObjectHandler2
Client
Confirm Action
ObjectHandler1
ObjectHandler2
Action
Server
invoke_action
(server_inst, action, information)
get handle number
start timer,
find target
ObjectHandler
action_request
(server_inst,
action,
information)
store
communication
partner
execute
(server_inst,
information)
action
(information)
action_request
(client_inst,
confirm_action,
reply)
invoke_action (client_inst,
confirm_action, reply)
release communication partner
confirm_action
(reply)
execute
(client_inst,
reply)
release handle
number
stop timer

FIG 1

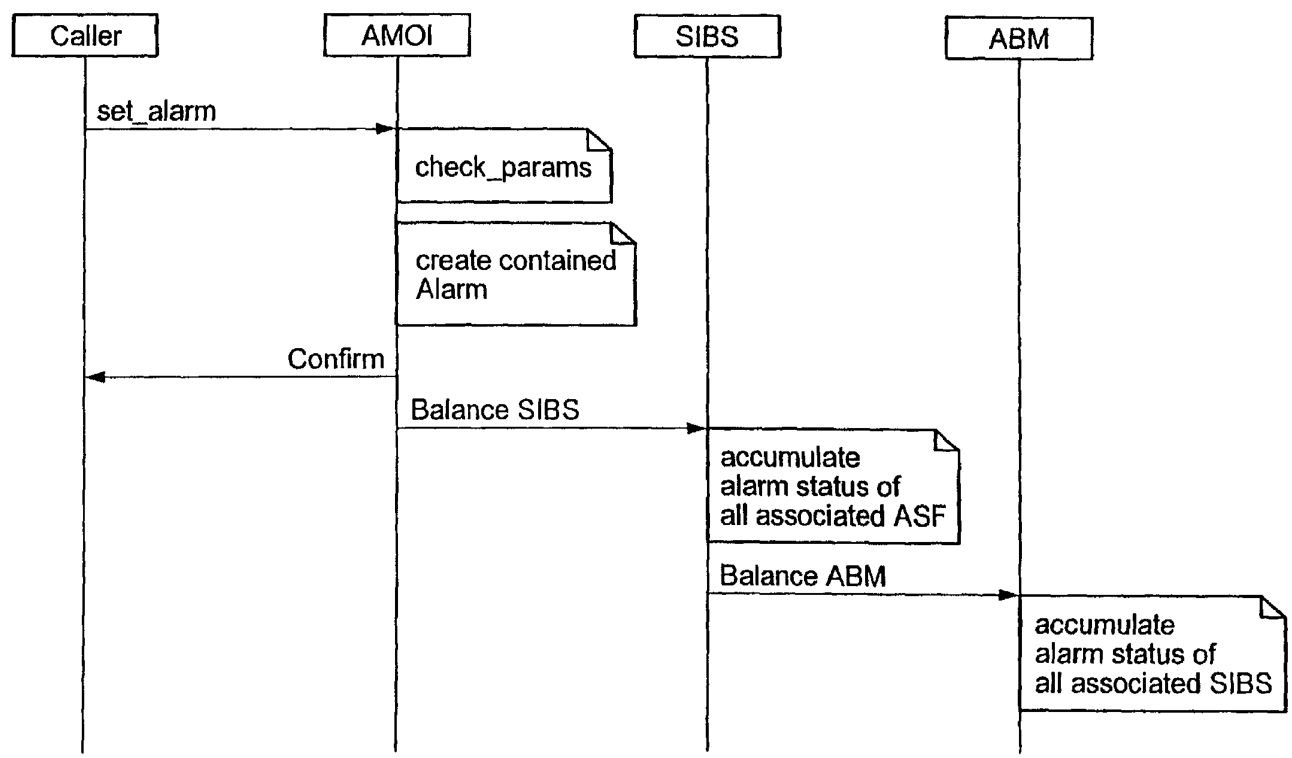

Caller
AMOI
SIBS
ABM
set_alarm
check_params
create contained
Alarm
Confirm
Balance SIBS
accumulate
alarm status of
all associated ASF
Balance ABM
accumulate
alarm status of
all associated SIBS

FIG 2

Declaration and Power of Attorney For Patent Application
Erklärung Für Patentanmeldungen Mit Vollmacht
German Language Declaration

Als nachstehend benannter Erfinder erkläre ich hiermit an Eides Statt:

dass mein Wohnsitz, meine Postanschrift, und meine Staatsangehörigkeit den im Nachstehenden nach meinem Namen aufgeführten Angaben entsprechen,

dass ich, nach bestem Wissen der ursprüngliche, erste und alleinige Erfinder (falls nachstehend nur ein Name angegeben ist) oder ein ursprünglicher, erster und Miterfinder (falls nachstehend mehrere Namen aufgeführt sind) des Gegenstandes bin, für den dieser Antrag gestellt wird und für den ein Patent beantragt wird für die Erfindung mit dem Titel:

Verfahren zur Nachrichtenübertragung zwischen einer einem ersten Prozess zugewiesenen Clientinstanz und wenigstens einer mindestens einem weiteren Prozess zugewiesenen Serverinstanz innerhalb eines verteilten Systems

deren Beschreibung

(zutreffendes ankreuzen)
☐ hier beigefügt ist.
☒ am 01.03.2000 als
PCT internationale Anmeldung
PCT Anmeldungsnummer PCT/DE00/00623
eingereicht wurde und am ______
abgeändert wurde (falls tatsächlich abgeändert).

Ich bestätige hiermit, dass ich den Inhalt der obigen Patentanmeldung einschliesslich der Ansprüche durchgesehen und verstanden habe, die eventuell durch einen Zusatzantrag wie oben erwahnt abgeändert wurde.

Ich erkenne meine Pflicht zur Offenbarung irgendwelcher Informationen, die für die Prüfung der vorliegenden Anmeldung in Einklang mit Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) von Wichtigkeit sind, an.

Ich beanspruche hiermit ausländische Prioritätsvorteile gemäss Abschnitt 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 119 aller unten angegebenen Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde, und habe auch alle Auslandsanmeldungen für ein Patent oder eine Erfindersurkunde nachstehend gekennzeichnet, die ein Anmeldedatum haben, das vor dem Anmeldedatum der Anmeldung liegt, für die Priorität beansprucht wird.

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

Method for transmitting, inside a distibuted system, messages between a client instance assigned to a first process and at least one server instance assigned to at least one additional process

the specification of which

(check one)
☐ is attached hereto.
☒ was filed on 01.03.2000 as
PCT international application
PCT Application No. PCT/DE00/00623
and was amended on ______
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

IDNR: 2590 / V: 99-1.00 / B:Val

German Language Declaration

Prior foreign appplications
Priorität beansprucht

Priority Claimed

Number / Nummer	Country / Land	Day Month Year Filed / Tag Monat Jahr eingereicht	Yes / Ja	No / Nein
19910345.3	DE	09.03.1999	☒	☐
(Number) (Nummer)	(Country) (Land)	(Day Month Year Filed) (Tag Monat Jahr eingereicht)	☐	☐
(Number) (Nummer)	(Country) (Land)	(Day Month Year Filed) (Tag Monat Jahr eingereicht)	☐	☐

Ich beanspruche hiermit gemäss Absatz 35 der Zivilprozessordnung der Vereinigten Staaten, Paragraph 120, den Vorzug aller unten aufgeführten Anmeldungen und falls der Gegenstand aus jedem Anspruch dieser Anmeldung nicht in einer früheren amerikanischen Patentanmeldung laut dem ersten Paragraphen des Absatzes 35 der Zivilprozeßordnung der Vereinigten Staaten, Paragraph 122 offenbart ist, erkenne ich gemäss Absatz 37, Bundesgesetzbuch, Paragraph 1.56(a) meine Pflicht zur Offenbarung von Informationen an, die zwischen dem Anmeldedatum der früheren Anmeldung und dem nationalen oder PCT internationalen Anmeldedatum dieser Anmeldung bekannt geworden sind.

I hereby claim the benefit under Title 35. United States Code. §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §122, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occured between the filing date of the prior application and the national or PCT international filing date of this application.

(Application Serial No.) (Anmeldeseriennummer)	(Filing Date D, M, Y) (Anmeldedatum T, M, J)	(Status) (patentiert, anhängig, aufgegeben)	(Status) (patented, pending, abandoned)
PCT/DE00/00623	01.03.2000	anhängig	pending

Ich erkläre hiermit, dass alle von mir in der vorliegenden Erklärung gemachten Angaben nach meinem besten Wissen und Gewissen der vollen Wahrheit entsprechen, und dass ich diese eidesstattliche Erklärung in Kenntnis dessen abgebe, dass wissentlich und vorsätzlich falsche Angaben gemäss Paragraph 1001, Absatz 18 der Zivilprozessordnung der Vereinigten Staaten von Amerika mit Geldstrafe belegt und/oder Gefängnis bestraft werden koennen, und dass derartig wissentlich und vorsätzlich falsche Angaben die Gültigkeit der vorliegenden Patentanmeldung oder eines darauf erteilten Patentes gefährden können.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

German Language Declaration

VERTRETUNGSVOLLMACHT: Als benannter Erfinder beauftrage ich hiermit den nachstehend benannten Patentanwalt (oder die nachstehend benannten Patentanwälte) und/oder Patent-Agenten mit der Verfolgung der vorliegenden Patentanmeldung sowie mit der Abwicklung aller damit verbundenen Geschäfte vor dem Patent- und Warenzeichenamt: *(Name und Registrationsnummer anführen)*

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith. *(list name and registration number)*

Customer No. 29177

29177

PATENT TRADEMARK OFFICE

And I hereby appoint

Telefongespräche bitte richten an: *(Name und Telefonnummer)*

Direct Telephone Calls to: *(name and telephone number)*

Ext. ______________

Postanschrift:

Send Correspondence to:

Bell, Boyd & Lloyd LLC
Three First National Plaza, 70 West Madison Street, Suite 3300 60602-4207 Chicago, Illinois
Telephone: (001) 312 372 11 21 and Facsimile (001) 312 372 20 98
or
Customer No. 29177

Voller Name des einzigen ~~oder ursprünglichen~~ Erfinders: Dr. MICHAEL WAGNER	Full name of sole or first inventor: Dr. MICHAEL WAGNER
Unterschrift des Erfinders: Michael Wagner — Datum: 9.12.01	Inventor's signature — Date
Wohnsitz: MUENCHEN, DEUTSCHLAND	Residence: MUENCHEN, GERMANY
Staatsangehörigkeit: DE DEY	Citizenship: DE
Postanschrift: GLEIWITZER STR. 28	Post Office Addess: GLEIWITZER STR. 28
81929 MUENCHEN	81929 MUENCHEN
Voller Name des zweiten Miterfinders (falls zutreffend):	Full name of second joint inventor, if any:
Unterschrift des Erfinders — Datum	Second Inventor's signature — Date
Wohnsitz: ,	Residence: ,
Staatsangehörigkeit	Citizenship
Postanschrift	Post Office Address

(Bitte entsprechende Informationen und Unterschriften im Falle von dritten und weiteren Miterfindern angeben).

(Supply similar information and signature for third and subsequent joint inventors).